Parameterized Approximability of the Disjoint Cycle Problem

Martin Grohe and Magdalena Grüber

Institut für Informatik, Humboldt-Universität, Unter den Linden 6, 10099 Berlin, Germany. {grohe,grueber}@informatik.hu-berlin.de

Abstract. We give an fpt approximation algorithm for the directed vertex disjoint cycle problem. Given a directed graph G with n vertices and a positive integer k, the algorithm constructs a family of at least $k/\rho(k)$ disjoint cycles of G if the graph G has a family of at least k disjoint cycles (and otherwise may still produce a solution, or just report failure). Here ρ is a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. The running time of our algorithm is polynomial.
The directed vertex disjoint cycle problem is hard for the parameterized complexity class W[1], and to the best of our knowledge our algorithm is the first fpt approximation algorithm for a natural W[1]-hard problem.
Key words: approximation algorithms, fixed-parameter tractability, parameterized complexity theory.

1 Introduction

Fixed-parameter tractability and approximability are two ways of dealing with intractability. Fixed-parameter tractability is a relaxed notion of tractability based on a "two-dimensional" complexity measure that not only depends on the size of the input instances, but also on an additional parameter, for example, the tree width of a graph, or the number of variables of a logical formula. Intuitively, a problem is fixed-parameter tractable if it can be solved efficiently on instances with a small value of the parameter. Formally, a parameterized problem is *fixed parameter tractable* if there is an algorithm solving the problem with a running time $f(k) \cdot n^{O(1)}$, where f is a computable function, k denotes the parameter value and n the size of the input. We call an algorithm achieving such a running time an *fpt algorithm.* Optimization problems are often parameterized by the cost of the solution. For example, the *disjoint cycle problem*[1] asks for a maximum family of vertex disjoint cycles in a given directed graph. In the parameterized version, the objective is to find a family of at least k disjoint cycles, where k is the parameter. Parameterized complexity theory gives evidence for the disjoint cycle problem not being fixed-parameter tractable; the problem is hard for the complexity class W[1] (this follows easily from the results of [20]). Now we may ask if the problem is at least "approximately fixed parameter tractable", that is, if there is an fpt

[1] More precisely, we call the problem the *maximum directed vertex disjoint cycle problem.* It is also known as the *(maximum directed) cycle packing problem.*

algorithm that finds a family of disjoint cycles of size approximately k if the input graph has a family of k disjoint cycles. Depending on the desired approximation ratio, "approximately k" may mean, for example, $k/2$ or maybe just $\log k$.

The notion of parameterized approximability has been introduced in three independent papers that all appeared last year [3, 4, 6]; also see the recent survey [14]. An fpt algorithm for a parameterized maximization problem is an *fpt approximation algorithm* with *approximation ratio* ρ if given an instance of the problem and a positive integer k it produces a solution of cost at least $k/\rho(k)$ if the instance has a solution of size at least k. (If the instance has no solution of size k, then the output may be arbitrary.) Here ρ is a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. An analogous definition can be given for minimization problems. We observe that whenever a maximization problem has an fpt approximation algorithm with approximation ratio ρ, then it also has a polynomial time approximation algorithm, albeit with a worse ratio $\rho' \geq \rho$ (but still only depending on k). [4, 6] give several examples of optimization problems that are not fpt approximable (under standard assumptions from parameterized complexity theory). Moreover, it was shown in [4] that every intractable parameterized problem is equivalent, under suitable reductions, to an (artificial) problem that is fpt approximable and to a problem that is fpt inapproximable. However, there are few examples of natural problems that are known to be fpt approximable, but not known to be fixed-parameter tractable.

The main result of this paper is an fpt approximation algorithm for the disjoint cycle problem. To the best of our knowledge, this is the first fpt approximation algorithm for a natural W[1]-hard problem. Our result is mainly of theoretical interest, because the approximation ratio is very close to k: We can only lower bound the number $k/\rho(k)$ of disjoint cycles our algorithm is guaranteed to compute by a multiply iterated logarithm.

The disjoint cycle problem is well known to be NP-complete (see [2]) and recently, Salavatipour and Verstraete [18] proved that this problem is hard to approximate within a factor $\Omega(\log^{1-\epsilon} n)$ unless $\mathrm{NP} \subseteq \mathrm{DTIME}\big(2^{\mathrm{polylog}(n)}\big)$. They also proved that the problem has a polynomial time approximation algorithm with ratio $O(\sqrt{n})$. Note that this approximation algorithm is incomparable with ours, as the ratio is expressed in terms of n, which may be much larger than k.

The linear programming dual of the disjoint cycle problem is the directed feedback vertex set problem. It asks for a minimum set of vertices of a directed graph such that the graph obtained by deleting these vertices is acyclic. It is one of the most prominent open problems in parameterized complexity theory whether the directed feedback vertex set problem is fixed-parameter tractable. Based on a result by Seymour [19], Even et al. [8] gave a polynomial time algorithm that approximates the minimum feedback vertex set to a factor of $O(\log k \cdot \log\log k)$; hence in particular the problem is fpt approximable. The size $\tau(G)$ of the minimum feedback vertex set and the size $\nu(G)$ of a maximum family of disjoint cycles are two graph invariants that have also received considerable attention in graph theory (e.g. [1, 2, 7, 12, 13, 17]). Clearly, $\nu(G) \leq \tau(G)$, and inequality may hold (for example, $\nu(K_4) = 1 < 2 = \tau(K_4)$). In 1996, Reed, Robertson, Seymour and Thomas [16] proved that $\tau(G)$ can be bounded in terms of $\nu(G)$; this settled a long standing open conjecture due to Younger [21]. As $\tau(G)$ can be approximated up to a logarithmic factor in polynomial time, it follows that there is a polynomial time algorithm that approximates $\nu(G)$ with some approximation ratio ρ (not depending on the size of the graph; see [4] for details). However, this algorithm just computes an approximation of the solution size and not an actual solution, that is, a family of pairwise disjoint cycles of that size. This is what we achieve here. Let us remark that in the last section of [16], Reed et al. studied the algorithmic question of computing a maximum family of disjoint cycles and gave an algorithm that computes a family of k disjoint cycles in time $n^{f(k)}$, for some function f. But this algorithm is not an fpt algorithm.

Our algorithm is heavily based on [16]; indeed in large parts it may be viewed as an algorithmic version of the construction given there. However, there is a crucial step in the construction of [16] that cannot be made algorithmic so easily, or more precisely, cannot be turned into an fpt algorithm: Reed et al. start their construction by taking a minimum feedback vertex set. We do not know how to compute such a set by an fpt algorithm. Instead, we take an approximately minimum feedback vertex set, but then a rather straightforward argument by Reed et al. has to be turned into a complicated recursive algorithm. We will explain this algorithm in detail in Section 5. Due to space limitations, we have to omit most other details and proofs.

2 Preliminaries

$\mathbb{N}$ denotes the set of positive integers and $\mathbb{R}$ the set of reals. A function $f : \mathbb{N} \to \mathbb{R}$ is *nondecreasing* (*increasing*) if for all $m, n \in \mathbb{N}$ with $m < n$ it holds that $f(m) \leq f(n)$ ($f(m) < f(n)$, respectively). f is *unbounded* if for all $k \in \mathbb{N}$ there exists an $n \in \mathbb{N}$ such that $f(n) \geq k$.

For a graph G the set of vertices (edges) is denoted by $V(G)$ ($E(G)$) and the number of vertices is denoted by $|G|$. For a subset $X \subseteq V(G)$, by $G[X]$ we denote the induced subgraph of G with vertex set X, and by $G \setminus X$ the subgraph $G[V(G) \setminus X]$. Graphs are always directed. *Paths* are directed and have no repeated vertices, and an (a, b)*-path* in a graph G is a path from a to b. A *linkage* L in a directed graph G is a set of vertex disjoint paths. A linkage L consisting of paths $P_1, \dots P_k$, where P_i is an (a_i, b_i)-path ($1 \leq i \leq k$), is said to *link* $(a_1, \dots a_k)$ to $(b_1, \dots b_k)$. For $A, B \subseteq V(G)$ with $a_1, \dots a_k \in A$ and $b_1, \dots, b_k \in B$ we say that L is a linkage *from* A *to* B *of size* k. A pair (X, Y) is a *separation* for a directed graph G (of *order* $|X \cap Y|$) if $X, Y \subseteq V(G)$ with $X \cup Y = V(G)$ and if no edge of G has tail in $X \setminus Y$ and head in $Y \setminus X$. Menger's theorem states that for all directed graphs G, sets $A, B \subseteq V(G)$, and $k \geq 1$ there is a linkage of size k from A to B if and only if there is no separation (X, Y) of G of order $< k$ with $A \subseteq X$ and $B \subseteq Y$. We will use the following algorithmic version, which can be proved by standard network flow techniques:

Fact 1. *There exists a polynomial time algorithm that given a directed graph G, a positive integer k and two subsets $A, B \subseteq V(G)$ both of size $\geq k$ computes* *either k many vertex disjoint paths from A to B or a separation (X, Y) of order $< k$ with $A \subseteq X$, $B \subseteq Y$.*

The *Ramsey number* $R_l(m, q)$ is the minimum integer such that for any set of size $\geq R_l(m, q)$ if all subsets of size l are colored with q colors, then there exists a subset of size m such that all its size-l subsets have the same color. A simple upper bound is

$$R_l(m, q) \leq \exp^{(l)}(c_q \cdot m) \tag{1}$$

for some positive constant c_q, where the iterarted exponential function $\exp^{(l)}$ is defined inductively as $\exp^{(1)}(x) = x$ and $\exp^{(l)}(x) := 2^{\exp^{(\ell-1)}(x)}$ for $l \geq 2$ [10].

3 Disjoint Cycles and Feedback Vertex Sets

Recall that the *directed maximimum vertex disjoint cycle problem* (for short: *disjoint cycle problem*) is the maximization problem whose objective it is to find a family of pairwise vertex disjoint cycles of maximum size in a given directed graph. For a directed graph G, we denote the maximum size of a family of pairwise vertex disjoint cycles by $\nu(G)$.

The linear programming dual of the disjoint cycle problem is the *directed minimum feedback vertex set problem.* A *feedback vertex set* in a directed graph G is a set S of vertices such that the graph $G \setminus S$ is acyclic. The objective of the directed minimum feedback vertex set problem is to find a feedback vertex set of minimum size in a given graph. We denote the minimum size of a feedback vertex set of a directed graph G by $\tau(G)$. Obviously, we have $\nu(G) \leq \tau(G)$. For a given graph $G = (V, E)$, consider the following linear program with variables x_v for the vertices $v \in V$:

$$\begin{array}{llll} \text{Minimize } \sum\limits_{v \in V} x_v & \text{subject to } \sum\limits_{v \in C} x_v \geq 1 & \text{for every cycle } C \text{ in } \mathcal{G}, \\ & \qquad\quad x_v \geq 0 & \text{for every vertex } v \in V. \end{array} \tag{2}$$

The $\{0,1\}$-solutions correspond to the feedback vertex sets of G. The rational solutions are called *fractional feedback vertex sets*, and the minimum cost $\sum_{v \in V} x_v$ of a fractional feedback vertex set is denoted by $\tau^*(G)$. By standard techniques [11], a minimum fractional feedback vertex set and the parameter $\tau^*(G)$ can be computed in polynomial time (also see [8]). The natural linear programming formulation of the disjoint cycle problem yields the dual of the linear program (2). Hence the cost $\nu^*(G)$ of an optimal fractional solutions coincides with $\tau^*(G)$, and we have

$$\nu(G) \leq \nu^*(G) = \tau^*(G) \leq \tau(G).$$

Let us state the main known results regarding the parameters τ and ν:

Theorem 2 (Seymour [19], Even et al. [8]). *For every directed graph G it holds that $\tau(G) = O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log\log(\tau^*(G)))$. Furthermore, there is a polynomial time algorithm that constructs a feedback vertex set for a directed graph G of size at most $O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log\log(\tau^*(G)))$.*

Theorem 3 (Reed et al. [16]). *There exists a computable function f such that for all directed graphs G it holds that $\tau(G) \leq f(\nu(G))$.*

The function f constructed by Reed at al. [16] grows very quickly. It is a multiply iterated exponential, where the number of iterations is also a multiply iterated exponential.

4 Fixed Parameter Tractable Approximation Algorithms

For background in parameterized complexity theory, we refer the reader to [5, 9, 15]. We only define the notions needed here. Recall the definition of fpt algorithms from the introduction. Intuitively, an fpt approximation algorithm is an algorithm whose running time is fpt for the parameter "cost of the solution" and whose approximation ratio only depends on the parameter and not on the size of the input. Hence every polynomial time approximation algorithm with constant approximation ratio is an fpt approximation algorithm, but an approximation algorithm with approximation ratio $\log n$, where n denotes the input size, is not.

For simplicity, we only define fpt approximation algorithms for maximization problems. The definition can easily be adapted for minimization problems.

Definition 4. Let $\rho : \mathbb{N} \to \mathbb{N}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. An *fpt approximation algorithm* for an NP-maximization problem O (over some alphabet Σ) with *approximation ratio* ρ is an algorithm $\mathbb{A}$ with the following properties:

1. $\mathbb{A}$ expects inputs $(x, k) \in \Sigma^* \times \mathbb{N}$. For every input $(x, k) \in \Sigma^* \times \mathbb{N}$ such that there exists a solution for x of cost at least k, the algorithm $\mathbb{A}$ computes a solution for x of cost at least $k/\rho(k)$. For inputs $(x, k) \in \Sigma^* \times \mathbb{N}$ without solution of cost at least k, the output of $\mathbb{A}$ can be arbitrary.
2. There exists a computable function f such that the running time of $\mathbb{A}$ on input (x, k) is bounded by $f(k) \cdot |x|^{O(1)}$.

O is *fpt approximable* if there exists an fpt approximation algorithm for O with ratio ρ, for some computable function $\rho : \mathbb{N} \to \mathbb{N}$ such that $k/\rho(k)$ is nondecreasing and unbounded.

The following lemma, which may be surprising at first sight, but is actually quite simple, states that we can make the running time of an fpt approximation algorithm polynomial at the cost of a worse approximation ratio. Note that this is not something one would usually do in practice, but it is relevant theoretically because it shows that the notion of fpt approximability is fairly robust.

Lemma 5. *For every fpt approximable maximization problem O there exists a polynomial time algorithm $\mathbb{A}$ that is an fpt approximation algorithm for O.*

In other words, we can replace requirement (2) in Definition 4 by the stronger requirement (2') below and obtain an equivalent notion of fpt approximability:

(2') The running time of $\mathbb{A}$ on input (x, k) is $\big(|x| + k\big)^{O(1)}$.

Let us remark that the analogue of Lemma 5 for minimization problems does not hold.

We also need the following definition to establish a relationship between approximation algorithms that are efficient for small optima and fpt approximation algorithms. For an instance x of an NP-optimization problem O, by $\mathrm{opt}(x)$ we denote the cost of an optimal solution.

Definition 6. An NP-maximization problem O is *well-behaved for parameterized approximation* if there exists a constant $c \geq 0$ such that the following holds:

1. Given an instance $x \neq \epsilon$ for O it is possible to construct a new instance x' for O in time polynomial in $|x|$ such that $|x'| < |x|$ and $\mathrm{opt}(x) \geq \mathrm{opt}(x') \geq \mathrm{opt}(x) - c$. (Here we assume that the empty string ϵ is an instance for O.)
2. For every instance $x \neq \epsilon$ it holds that a valid solution for the constructed instance x' is also a valid solution for x.

For example, the disjoint cycle problem is well-behaved, because given a graph G we can delete a vertex w and obtain a graph $G' = G \setminus \{w\}$ such that $\nu(G) \geq \nu(G') \geq \nu(G) - 1$, and every family of disjoint cycles of G' is also a family of disjoint cycles of G.

Proposition 7 (Chen et al. [4]). *Let O be an* NP*-maximization problem over the alphabet Σ that is well-behaved for parameterized approximation, and let $\rho : \mathbb{N} \to \mathbb{N}$ be a computable function such that $k/\rho(k)$ is nondecreasing and unbounded. If there exists a computable function g and an algorithm $\mathbb{B}$ that given an input $x \in \Sigma^*$ computes a solution y for x of cost k such that $k \geq \mathrm{opt}(x)/\rho(\mathrm{opt}(x))$ in time $g(\mathrm{opt}(x)) \cdot |x|^{O(1)}$, then O has an fpt approximation algorithm with approximation ratio ρ.*

5 The Main Theorem

Theorem 8. *The directed maximum vertex disjoint cycle problem has an fpt approximation algorithm with polynomial running time.*

First, we will design an algorithm $\mathbb{A}$ that computes at least $\tau^*(G)/\rho(\tau^*(G))$ vertex disjoint cycles for a given directed graph G and some computable function $\rho : \mathbb{N} \to \mathbb{N}$ with $\frac{x}{\rho(x)}$ being nondecreasing and unbounded and such that the running time of this algorithm is bounded by $g(\tau^*(G)) \cdot |G|^{O(1)}$ for some computable function g. We will then see why this also gives an fpt approximation algorithm for the disjoint cycle problem.

The core of our algorithm $\mathbb{A}$ is a recursive procedure, described in Section 5.2, that either computes sufficiently many vertex disjoint cycles in a given graph G directly, or guarantees the existence of linkages between all subsets A, B of a certain size m of some feedback vertex set T of G. It remains to give an algorithm that computes sufficiently many vertex disjoint cycles if such linkages exist. This is what the "Linkage Lemma" (Lemma 10) achieves. Very roughly, the existence of the linkages guarantees the existence of a substructure of the graph called a "fence", together with a linkage that connects the "bottom" with the "top" of the fence. In such a fence it is then possible to find the desired cycles.

5.1 Technical Lemmas

Lemma 9 (Splitting Lemma). *Let G be a directed graph, T a feedback vertex set for G, and $m \leq |T|/2$. Then at least one of the following possibilities holds:*

(i) For all $A, B \subseteq T$ with $|A| = |B| = m$ there exists a linkage from A to B with no vertex in $T \setminus (A \cup B)$.
(ii) There is a feedback vertex set T' of G with $|T'| < |T|$.
(iii) There are two vertex disjoint subgraphs G_1 and G_2 of G, cycles C_1 in G_1 and C_2 in G_2, and feedback vertex sets T_1 for G_1 and T_2 for G_2 such that:
- *$|T_1| = |T_2| = m$.*
- *For all feedback vertex sets T_1' of G_1 and T_2' of G_2 the set $T_1' \cup T_2' \cup \big(V(G) \setminus V(G_1 \cup G_2)\big)$ is a feedback vertex set of G of size at most $|T_1'| + |T_2'| + |T| - (m+1)$.*

Furthermore, there is an algorithm that, given G, T, and m, either recognizes that (i) holds or computes a feedback vertex set T' as in (ii) or computes G_1, C_1, T_1, G_2, C_2, T_2 as in (iii). The running time of the algorithm is $3^{|T|} \cdot |G|^{O(1)}$.

Proof. Suppose that (i) does not hold. Let $A, B \subseteq T$ with $|A| = |B| = m$ such that there is no linkage from A to B in G that has no vertex in $Z := T \setminus (A \cup B)$. By Menger's theorem, there exists a separation (X, Y) of G with $A \subseteq X$, $B \subseteq Y$, $Z \subseteq (X \cap Y)$ and with $|W| < m$, where $W := X \cap Y \setminus Z$. Let G_1 be $G \setminus Y$ and

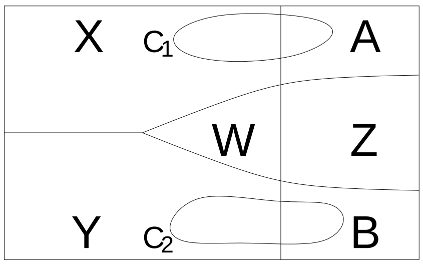

Fig. 1. A sketch of the separation (X, Y) for G (assuming $X \cap B = \emptyset$ and $Y \cap A = \emptyset$)

G_2 be $G \setminus X$. Furthermore let $D_1 := A \cup (X \cap Y)$ and $D_2 := B \cup (X \cap Y)$.

Then it holds that $|D_1| = |D_2| < |T|$, because $X \cap Y = W \cup Z$, $|Z| = |T \setminus (A \cup B)| = |T| - 2m$, and $|W| < m$. It might happen that D_1 or D_2 is again a feedback vertex set for G, in which case we succeeded at finding a smaller feedback vertex set for G. Thus (ii) holds. Otherwise, there are cycles C_1, C_2 such that $V(C_1) \cap D_2 = \emptyset$ and $V(C_2) \cap D_1 = \emptyset$. We claim that $C_i \subseteq G_i$ for $i = 1, 2$. Let us look at C_1; the proof for C_2 is completely analogous. Since T is a feedback vertex set, we have $T \cap V(C_1) \neq \emptyset$ and thus $A \cap V(C_1) \neq \emptyset$, because $D_2 \cap V(C_1) = \emptyset$. Since (X, Y) is a separation and $X \cap Y \cap V(C_1) = \emptyset$, it follows that $V(C_1)$ does not meet Y and therefore the cycle C_1 is contained in G_1.

Observe that A and B are feedback vertex sets for G_1 and G_2, respectively, because every cycle must meet T, and $T \cap V(G_1) = A$. Hence we can let $T_1 := A$ and $T_2 := B$. It is easy to check that (iii) is satisfied. It remains to prove the algorithmic statement. Using Fact 1, we can compute sets A, B and a separation (X, Y) with the properties above if such sets A, B exist in time $3^{|T|} \cdot |G|^{O(1)}$. Here we use $3^{|T|}$ as an upper bound for the number of partitions of T into three sets A, B, Z. Given A, B, X, Y, we can check in polynomial time if the sets D_1, D_2 defined above are feedback vertex sets and find the cycles C_1, C_2 if not. $\square$

We define $\kappa = \kappa(G)$ for a given graph G to be the maximum integer with

$$(\kappa-1)^2+1+\exp^{((\kappa-1)^2+1)}(c_{(((\kappa-1)^2+1)!+1)^2}\cdot(\kappa+1)\cdot((\kappa-1)^2+1)) \leq \tau^*(G) \quad (3)$$

and let

$$\lambda(G) := (\kappa(G)-1)^2+1 \qquad \text{and} \qquad \mu(G) := (\kappa(G)+1)\cdot\lambda(G) \ . \quad (4)$$

Lemma 10 (Linkage Lemma). *There is a computable, nondecreasing and unbounded function φ such that the following holds: Let G be a directed graph, $m \leq \mu(G)$, and let T be a feedback vertex set of G such that for all $A, B \subseteq T$ with $|A| = |B| = m$ there exists a linkage from A to B with no vertex in $T \setminus (A \cup B)$. Then there exists a family of at least $\varphi(m)$ vertex disjoint cycles in G. Furthermore, there is an algorithm that computes such a family in time $g(|T|) \cdot |G|^{O(1)}$, for some computable function g.*

The proof of this lemma is based on algorithmic versions of Theorems 2.2., 2.3. and 2.4. of [16]. It will appear in the full version of this paper.

5.2 The Main Algorithm

Lemma 11. *There is a computable, nondecreasing, unbounded function ψ, a computable function g, and an algorithm $\mathbb{A}$ such that the following holds: Given a directed graph G, the algorithm $\mathbb{A}$ computes a family of at least $\psi(\tau^*(G))$ vertex disjoint cycles of G in time $g(\tau^*(G)) \cdot |G|^{O(1)}$*

Proof. Let G be a directed graph. Set $m_i := \lfloor \mu(G)/4^i \rfloor$ for $i \geq 0$ and let i^* be the minimal integer such that $m_i \leq 4$. Let φ be the function from the Linkage Lemma (Lemma 10). We define a function χ by

$$\chi(0) := \chi(1) := \min\{\varphi(1), i^*\}, \quad \chi(k) := \min\left\{\chi\left(\left\lfloor\frac{k}{4}\right\rfloor\right)+1, \varphi(k), i^*\right\}.$$

It is easy to see that χ is computable, nondecreasing, and unbounded. By definition of $\mu(G)$ in (4), there is a computable, nondecreasing and unbounded function ζ such that $\zeta(\tau^*(G)) = \mu(G)$ for all directed graphs G. We define the function ψ by $\psi(x) := \chi(\zeta(x))$.

On input G, algorithm $\mathbb{A}$ first computes a feedback vertex set T of G of size $O(\tau^*(G) \cdot \log \tau^*(G) \cdot \log \log \tau^*(G))$ using the algorithm of Theorem 2. Then $\mathbb{A}$ builds up a binary tree $\mathcal{B}$. The vertices of this tree are labelled by pairs (G', T'), where G' is a graph and T' a feedback vertex set of G'. The edges of the tree $\mathcal{B}$ are labelled by cycles of G.

At each point during the execution, the algorithm $\mathbb{A}$ processes a leaf of the tree and then either halts or modifies the tree and continues with some other node. If we think of the algorithm $\mathbb{A}$ as a recursive algorithm, then $\mathcal{B}$ is just a convenient way to describe the content of the stack during the execution of $\mathbb{A}$.

Initially, the tree $\mathcal{B}$ only consists of its root, which is labelled by the pair (G, T). The algorithm starts at this root. Now suppose at some stage during its execution the algorithm has to process a leaf b labelled by (G_b, T_b) and of height i. (The *height* of b is defined to be the length of the path from the root to b.) If $i \geq i^*$, the algorithm $\mathbb{A}$ halts and outputs the cycles labelling the edges of the path from the root to b. Otherwise, it calls the algorithm of the Splitting Lemma (Lemma 9) with input G_b, T_b, and m_i. (We will prove later that the assumption $2m_i \leq T_b$ is satisfied.) According to the three outcomes of the Splitting Lemma, we have to distinguish the following cases:

Linkage Step. If for all $A, B \subseteq T_b$ with $|A| = |B| = m_i$ there exists a linkage in G_b from A to B with no vertex in $T_b \setminus (A \cup B)$, then the algorithm of the Linkage Lemma is called with input (G_b, T_b). It returns a family $\mathcal{C}$ of $\varphi(m_i)$ pairwise disjoint cycles of G_b. Algorithm $\mathbb{A}$ halts and outputs the cycles in $\mathcal{C}$ together with the cycles labelling the edges of the path from the root to b in $\mathcal{B}$.

Splitting Step. If the algorithm of the Splitting Lemma returns vertex disjoint subgraphs G_1, G_2, cycles $C_1 \subseteq G_1$ and $C_2 \subseteq G_2$, and feedback vertex sets T_1 of G_1 and T_2 of G_2, then our algorithm $\mathbb{A}$ proceeds as follows: It creates two new children b_1, b_2 of b. For $i = 1, 2$, child b_i is labelled (G_i, T_i), and the edge from b to b_i is labelled by the cycle C_{2-i} (so that the cycle labelling the edge to b_i is disjoint from the graph labelling the node b_i). Now the processing of b is completed, and algorithm $\mathbb{A}$ continues with child b_1.

Shrinking Step. If the algorithm of the Splitting Lemma returns a feedback vertex set T_b' of G_b with $|T_b'| < |T_b|$, then our algorithm $\mathbb{A}$ relabels the current leaf b by (G_b, T_b'). Then it calls the following subroutine $\mathbb{S}$ at leaf b.

Subroutine $\mathbb{S}$. The subroutine $\mathbb{S}$ proceeds as follows. Suppose it is at a leaf b_1 of $\mathcal{B}$ labelled (G_1, T_1). If b_1 is the root, the subroutine returns b_1. Otherwise, let b' be the parent and b_2 the sibling of b_1. Suppose that b' is labelled (G', T') and b_2 is labelled (G_2, T_2). Let $T'' = T_1 \cup T_2 \cup \big(V(G') \setminus (V(G_1 \cup G_2)\big)$. $\mathbb{S}$ distinguishes between three cases:

– If $|T''| < |T'|$, then $\mathbb{S}$ deletes b_1 and b_2. Then it relabels b' by (G', T'') and continues at node b'.

– If $|T''| \geq |T'|$ and $|T_2| > |T_1|$, then $\mathbb{S}$ returns b_2.
– Otherwise, $\mathbb{S}$ returns b_1.

Now $\mathbb{A}$ continues with the leaf returned by $\mathbb{S}$.

It is easy to check that the node b processed by the algorithm is always a leaf (as required). To see this, note that whenever the algorithm processes a node b, the sibling of every node on the path from the root to b is a leaf. (Hence actually the tree $\mathcal{B}$ is always quite degenerate.)

Furthermore, using the Splitting Lemma, it is easy to show that the following invariants are maintained throughout the execution of the algorithm:

(1) Let b be a node of the tree labelled (G_b, T_b). Then the cycles labelling the edges on a path from the root to b are pairwise disjoint, and they are all disjoint from the graph G_b.
(2) Let b be a node of the tree labelled (G_b, T_b). Then T_b is a feedback vertex set of G_b.
(3) Let b_1 be a node labelled (G_1, T_1) of height $i+1$, for some $i \geq 0$. Suppose that the parent b' is labelled (G', T') and that the sibling b_2 of b is labelled (G_2, T_2). Then G_1 and G_2 are vertex disjoint subgraphs of G. For all feedback vertex sets T_1' of G_1 and T_2' of G_2 the set $T_1' \cup T_2' \cup \big(V(G) \setminus V(G_1 \cup G_2)\big)$ is a feedback vertex set of G of size at most $|T_1'| + |T_2'| + |T'| - (m_i + 1)$. Furthermore, it holds that $m_i < |T_1| + |T_2| \leq 2m_i$.

For the lower bound $m_i < |T_1| + |T_2|$ in (3) note that as soon as $|T_1| + |T_2| \leq m_i$ for some children b_1, b_2 of b' labelled (G_1, T_1) and (G_2, T_2) it holds that $|T''| < |T'|$ for $T'' := T_1' \cup T_2' \cup \big(V(G) \setminus V(G_1 \cup G_2)\big)$ and the subroutine $\mathbb{S}$ deletes b_1 and b_2.

(4) Suppose that at some point during the execution a leaf b_1 labelled (G_1, T_1) is processed while the sibling b_2 of b is labelled (G_2, T_2). Then $|T_1| \geq |T_2|$.

(3) and (4) imply:

(5) Suppose that at some point during the execution of $\mathbb{A}$ a leaf b_1 labelled (G_1, T_1) and of height $i+1$, for some $i \geq 0$, is processed. Then $m_i/2 \leq |T_1|$.

Finally, note that throughout the execution:

(6) The root is labelled (G, T_0) for some feedback vertex set T_0 of size $2m_0 \leq |T_0| = O(\tau^*(G) \cdot \log \tau^*(G) \cdot \log\log \tau^*(G))$.

The first inequality in (6) holds as $2m_0 = 2\mu(G) \leq \tau^*(G) \leq \tau(G)$ by the definition of μ. Since $m_{i+1} \leq m_i/4$, it follows that throughout the execution:

(7) Whenever a node b of height i labelled (G_b, T_b) is processed, we have $2m_i \leq |T_b|$.

This justifies the application of the Splitting Lemma in the algorithm.

Observe that (1) implies that whenever the algorithm halts, its output is a family of disjoint cycles. The size of this family is i^* if the algorithm halts at a leaf of height i^*, or $i + \varphi(m_i)$, if the algorithm halts at a leaf of height $i < i^*$ in a Linkage Step. As $\chi(\mu(G)) = \chi(m_0) \leq \min\{i^*, i + \varphi(m_i)\}$ for $0 \leq i < i^*$ and as $\mu(G) = \zeta(\tau^*(G))$, these families of cycles are sufficiently large.

It remains to prove that the execution always terminates within the desired running time bounds. To do this, we upperbound the number N of shrinking steps that might occur while building the tree. For the root of the tree at most $|T| - \tau^*(G)$ shrinking steps can occur before we end up with a minimal feedback vertex set for the given graph G. For all nodes b' of height $i \geq 1$ we claim that at most $m_i \leq \mu(G)$ shrinking steps on children of b' can occur before the feedback vertex set for b' is shrunk and the children are deleted in subroutine $\mathbb{S}$. To see this, let b' be a node labelled (G', T') and of height i that has children b_1, b_2 labelled (G_1, T_1) and (G_2, T_2). Recall that by (3), the set $T'' := T_1 \cup T_2 \cup \big(V(G) \setminus V(G_1 \cup G_2)\big)$ is a feedback vertex set of G' of size at most $|T_1| + |T_2| + |T'| - (m_i + 1)$. Furthermore, it holds that $|T_1| + |T_2| \leq 2m_i$. So after at most m_i shrinking steps for the children of b it holds that b_1, b_2 are labelled with (G_1, T_1) and (G_2, T_2) such that $|T_1| + |T_2| \leq m_i$ and hence $|T''| < |T'|$. This implies that b' is relabelled by (G', T'') for this smaller feedback vertex set T'' in subroutine $\mathbb{S}$ after at most m_i shrinking steps on b_1, b_2.

As the tree $\mathcal{B}$ always contains at most two nodes with height i for $1 \leq i \leq i^*$ and as $m_i \leq m_0 = \mu(G)$ for all $1 \leq i \leq i^*$, there occur at most $(|T| - \tau^*(G)) \cdot \mu(G)^{i^*}$ shrinking steps in total. So by the definition of $\mu(G)$ and i^* and as $|T| = O(\tau^*(G) \cdot \log(\tau^*(G)) \cdot \log\log(\tau^*(G))$, the number N can be upper bounded by a function of $\tau^*(G)$. This implies that we can execute the algorithm by using the Splitting Lemma at most $2i^* \cdot (N+1)$ times and the Linkage Lemma at most once. In total, this results in a running time that is bounded by $g(\tau^*(G)) \cdot |G|^{O(1)}$ for some computable function g. $\square$

Proof (Proof of Theorem 8). The fpt approximability of the disjoint cycle problem follows from Lemma 11, Theorem 3, and Proposition 7, using the observation that the disjoint cycle problem is well-behaved for parameterized approximation. The polynomial running time can be obtained by Lemma 5. $\square$

6 Concluding remarks

We give an fpt approximation algorithm for the directed vertex disjoint cycle problem. We mainly see this result as a contribution to the evolving theory of parameterized approximability [3, 4, 6, 14], as it provides the first natural example of a parameterized problem that is fpt approximable, but known to be hard to be solved exactly (W[1]-hard, to be precise).

Our algorithm is based on the connection between disjoint cycles and feedback vertex sets and a structure theory for directed graphs developed in this context by Reed et al. [16]. To establish our result, we had to make large parts of this structure theory algorithmic. This may turn out to be useful in other contexts. In particular, it may help to find an fpt algorithm for the directed feedback vertex set problem and hence solve one of the most prominent open problems in parameterized complexity theory.

References

1. N. Alon. Disjoint directed cycles. *Journal of Combinatorial Theory Series B*, 68(2):167–178, 1996.
2. J. Bang-Jensen and G. Gutin. *Digraphs.* Springer-Verlag, 2002.
3. L. Cai and X. Huang. Fixed-parameter approximation: Conceptual framework and approximability results. In H. L. Bodlaender and M. A. Langston, editors, *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *LNCS*, pages 96–108. Springer-Verlag, 2006.
4. Y. Chen, M. Grohe, and M. Grüber. On parameterized approximability. In H. L. Bodlaender and M. A. Langston, editors, *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *LNCS*, pages 109–120. Springer-Verlag, 2006.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer-Verlag, 1999.
6. R. G. Downey, M. R. Fellows, and C. McCartin. Parameterized approximation algorithms. In H. L. Bodlaender and M. A. Langston, editors, *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation*, volume 4169 of *LNCS*, pages 121–129. Springer-Verlag, 2006.
7. P. Erdös and L. Pósa. On the independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.
8. G. Even, J. S. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.
9. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer-Verlag, 2006.
10. R. L. Graham, M. Grötschel, and L. Lovász, editors. *Handbook of Combinatorics*, volume II, chapter Ramsey theory, pages 1331–1403. Elsevier Science, 1995.
11. M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, 2nd edition, 1993.
12. B. Guenin and R. Thomas. Packing directed circuits exactly. 2001. To appear in *Combinatorica.*
13. G. Gutin and A. Yeo. Some parameterized problems on digraphs. 2006. Submitted.
14. D. Marx. Parameterized complexity and approximation algorithms. 2006. To appear in *The Computer Journal.*
15. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.
16. B. Reed, N. Robertson, P. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.
17. B. Reed and F. Shepherd. The gallai-younger conjecture for planar graphs. *Combinatorica*, 16(4):555–566, 1996.
18. M. Salavatipour and J. Verstraete. Disjoint cycles: Integrality gap, hardness, and approximation. In M. Jünger and V. Kaibel, editors, *Proceedings of the 11th International Conference on Integer Programming and Combinatorial Optimization*, volume 3509 of *LNCS*, pages 51–65. Springer-Verlag, 2005.
19. P. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.
20. A. Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In G. D. Battista and U. Zwick, editors, *Proceedings of the 11th Annual European Symposium on Algorithms, ESA '03*, volume 2832 of *LNCS*, pages 482–493, 2003.
21. D. Younger. Graphs with interlinked directed circuits. In *Proceedings of the Midwest Symposium on Circuit Theory 2*, pages XVI 2.1–XVI 2.7, 1973.